2º CONGRESSO BRASILEIRO DE P&D EM PETRÓLEO & GÁS

CONCEITOS BÁSICOS E TÉCNICAS DE PROCESSAMENTO PARALELO EM UM CLUSTER DE COMPUTADORES PARA DETERMINAÇÃO DE PROPRIEDADES FÍSICAS DE ROCHAS RESERVATÓRIO

Bueno A.D[1]

[1]*Laboratório de Engenharia e Exploração de Petróleo - LENEP*
Universidade Estadual do Norte Fluminense - CEP 27925-031- bueno@lenep.uenf.br

Resumo - A determinação de propriedades físicas de rochas reservatório utilizando-se a analise de imagens é uma realidade. A idéia de um laboratório virtual onde as propriedades petrofísicas podem ser determinadas combinando-se a realização de ensaios e a simulação numérica, tem se demonstrado uma tecnologia importante na redução dos custos e no aumento da velocidade com que as informações podem ser obtidas. Entretanto, algumas lâminas apresentam, após a binarização, um comprimento de correlação elevado, o que implica na necessidade de representações tridimensionais de elevada dimensão. Isto remete os desenvolvedores de modelos e algoritmos da área de análise de imagens de rochas reservatório a utilização dos mecanismos de processamento paralelo. Neste sentido, este artigo apresenta uma *breve revisão bibliográfica dos termos e conceitos necessários ao entendimento e uso do processamento paralelo e de cluster de computadores*. Descreve-se os *diferentes tipos de processamento paralelo* em máquinas multi-processadas. A seguir, apresenta-se as *bibliotecas* para implementação de *multi-processos, multi-threads*, e os sistemas de troca de mensagens como *PVM* e *MPI*. No final do artigo apresenta-se os tempos de processamento obtidos em testes realizados usando-se múltiplos processos e múltiplas threads em um cluster com 3 nós, e inclui-se um *conjunto de links e referências para documentos externos*.

Palavras-chave: cluster, processamento paralelo, análise de imagem, propriedades físicas de rochas reservatório.

Abstract - The determination of physical properties of reservoir rocks using image analysis is a reality. The idea of a virtual laboratory where the petro-physics properties are determined with experimental samples and numerical simulations, has demonstrated an important technology in the reduction of the costs and the increase of the speed with that the information are obtained. However, most samples, after the binarization, have a large correlation length, this implies in necessity of large 3D reconstructed image. This remit the developers of models and algorithm's of image analysis of reservoir rocks to use and understand the parallel processing and clusters of workstations. This paper present a brief bibliography revision of the terms and concepts of parallel processing. The deferents types of parallel processing in multi-processed machines are discussed. After this, the library's used´s are rapidly presented, multi-processing, multi-threading, and the message passing systens (PVM and MPI). Finally, a study case, comparing the time processing between multi-processing and multi-threading, in a cluster of 3 nodes, are presented. A set of links and references for external documents are included.

Keywords: cluster, parallel processing, image analysis, physical properties of reservoir rocks.

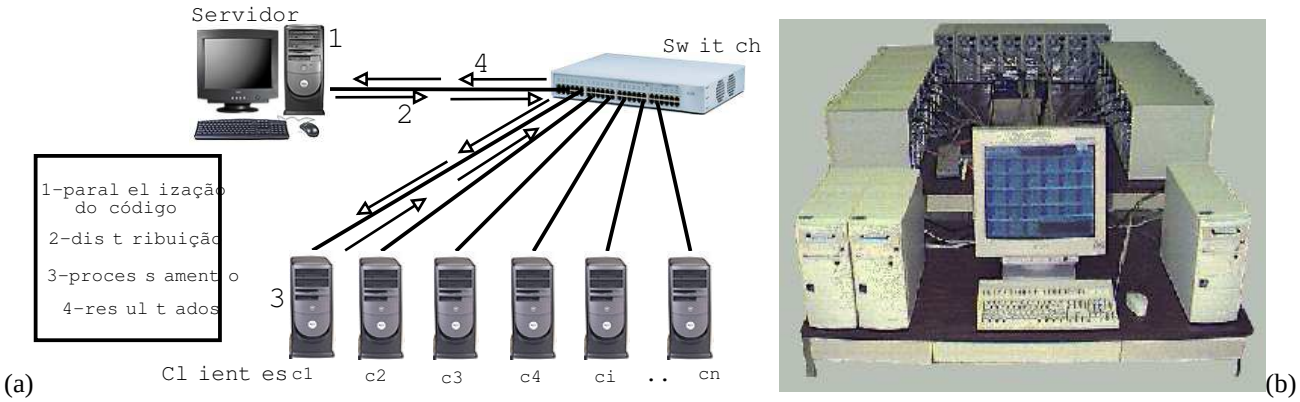


Figura 1: Em (a) diagrama de um cluster. Em (b) Cluster do Numerical Aerospace Simulation Facility NASA Ames.

1. Introdução

A determinação de propriedades físicas dos materiais com uso das ferramentas da análise de imagens (Gonzales e Woods (1993); Gomes e Velho (1994)), tem sido realizada e estudada por diversos autores (Coster e Chermant (1989); Adler *et al.*(1990)). Como exemplo, pode-se citar o uso dos modelos de gás em rede, Santos *et al.*(2001), que tem se demonstrado uma alternativa mais eficiente que os modelos de volumes finitos na determinação de propriedades físicas de meios porosos (por apresentarem estes uma geometria complexa). Outro exemplo é visto em Bueno e Philippi (2002). Segundo os autores, algumas lâminas apresentam comprimento de correlação elevado, o que implica na necessidade de se obter representações tridimensionais de elevada dimensão. A obtenção de representações 3D de elevada dimensão, com o método da gaussiana truncada e com o método da gaussiana truncada revisada (Bueno *et al.*(2002)), requerem computadores com elevada capacidade de memória. Como estes modelos são massissamente paralelizáveis, pode-se, com o uso de um cluster de computadores, aumentar o desempenho dos algoritmos, possibilitando o trabalho com representações tridimensionais de elevada dimensão.

Com o objetivo de possibilitar a obtenção de representações tridimensionais de elevada dimensão com o método da gaussiana truncada, bem como reduzir os tempos de processamento e o custo na determinação das propriedades físicas de rochas reservatório, pode-se utilizar mecanismos de processamento paralelo, tanto em máquinas com múltiplos processadores como em um cluster de computadores.

Deve-se considerar ainda que diversos grupos de pesquisa em engenharia de petróleo tem solicitado aos orgãos de financiamento (CNPQ) recursos para montagem e implementação de "cluster de computadores". Entretanto, o uso efetivo de um cluster de computadores envolve aspectos como: o conhecimento dos diferentes tipos de cluster e de processamento paralelo, seu funcionamento, o aprendizado de técnicas de paralelização de algoritmos e noções básicas das diferentes bibliotecas, Wilkinson e Allen (1999). O objetivo do presente artigo é apresentar uma breve revisão bibliográfica destes conceitos, fornecendo aos engenheiros, uma base conceitual para o desenvolvimento de algoritmos e programas em engenharia de petróleo que rodem em um cluster de computadores.

2. Clusters de computadores

O que é um cluster de computadores ? Um cluster de computadores é um conjunto de computadores (heterogêneos ou não) conectadas em rede para o desenvolvimento de processamento paralelo. Ou seja, as máquinas são conectadas via rede para formar um "*único computador*", Manika (1999). Veja o diagrama de um cluster na Figura 1 (a).

O objetivo de um cluster de computadores é possibilitar o uso de computadores ligados em rede para execução de processamento com alto desempenho, permitindo a realização de simulações avançadas. O projeto pioneiro em clusters de computadores, com o nome Beowulf, foi desenvolvido no CESDIS (Center of Excellence in Space Data and Information Sciences) em 1994 e contava com 16 máquinas 486 rodando GNU/Linux.

O **princípio de funcionamento** é simples.

▹**distribuição:** O servidor divide as tarefas em suas partes independentes (ditas concorrentes) e distribuí para os clientes. Observe na Figura 1 que o servidor envia e recebe mensagens (e dados) para os diversos clientes passando pelo switch.

▹**processamento e troca de mensagens entre clientes:** Cada cliente recebe as mensagens e um conjunto de dados a serem processados. A seguir realiza o processamento solicitado. Observe que podem ocorrer trocas de mensagens entre os diversos clientes.

▹**conclusão:** Concluído o processamento, os resultados são enviados para o servidor, que agrupa os resultados e finaliza o processamento.

Este tipo de processamento é conhecido como processamento paralelo distribuído. Para que o sistema funcione é necessário um servidor, vários clientes, uma biblioteca para troca de mensagens[1] entre o servidor e os clientes e o hardware[2] para conexão via rede dos diversos computadores.

[1]As bibliotecas mais usuais usadas em clusters de computadores são a PVM e a MPI, podendo-se ainda utilizar múltiplos processos que se comunicam com pipes.

[2]Observe que o hardware da rede (placas de rede, switch, cabos) deve ter qualidade e capacidade para transferir os dados do servidor para os clientes com a menor perda de tempo possível (latência).

Tipo 1->Cluster Beowulf: Beowulf é uma tecnologia de cluster que agrupa computadores rodando GNU/Linux para formar um supercomputador virtual via processamento paralelo (distribuído). Veja maiores detalhes em Donald J.Becker (1995), Daniel Ridge e Merkey (1997) e Radajewski e Eadline (1998). Veja na Figura 1 (b) um exemplo de cluster tipo Beowulf.

Requisitos: Conjunto de computadores (sem teclado, sem monitor e sem mouse) conectados em rede para processamento paralelo (uso exclusivo). Requer o uso de uma biblioteca de troca de mensagens como PVM ou MPI, ou o uso de múltiplos processos com o OpenMosix[3].

Vantagens: Manutenção facilitada, redução do número de problemas ocasionados pela instalação de pacotes desnecessários. Menor custo das máquinas e de manutenção.

Desvantagens: As máquinas tem seu uso limitado ao processamento definido pelo servidor.

Tipo 2->Cluster de "workstacions": Um cluster de "workstacions" é um conjunto de computadores completos (com teclado, monitor, mouse), conectados em rede, e que cumprem duas funções; o uso diário, com diversos tipos de programas como processadores de texto e planilhas, e o uso para processamento paralelo pesado no final do dia e/ou nos fins de semana. A Figura 2 (b) mostra um exemplo de cluster de computadores.

Requisitos: As máquinas devem ser completas e independentes. Requer o uso de uma biblioteca de troca de mensagens como PVM ou MPI, ou o uso de múltiplos processos com o OpenMosix.

Vantagens: Possibilita o uso das máquinas por diferentes usuários para realização de suas tarefas rotineiras.

Desvantagens: Como vários usuários estão utilizando os processadores para outras tarefas, o desempenho do sistema é reduzido. Na prática reduz o uso do cluster ao final do dia e nos fins de semana. Tem um custo maior por máquina e maiores problemas com a manutenção do sistema.

3. Objetivo, conceito e tipos de processamento paralelo

O **processamento paralelo** consiste em dividir uma tarefa em suas partes independentes e na execução de cada uma destas partes em diferentes processadores.

Basicamente é necessário: i) paralelizar os algoritmos, ii) um mecanismo para distribuição do processamento pelos diversos processadores disponíveis e iii) um mecanismo para troca de mensagens (informações e dados) entre os diferentes processos.

Existem diversos métodos e técnicas para implementar o processamento paralelo. Como exemplo, pode-se citar a divisão dos processos tendo como base as tarefas a serem realizadas (cada processador poderia realizar tarefas diferentes) ou a divisão dos processos utilizando-se a divisão do domínio (como exemplo a divisão de uma matriz). Maiores detalhes são abordados em Wilkinson e Allen (1999).

Descreve-se a seguir alguns dos diferentes tipos de estruturas utilizadas para implementar o processamento paralelo.

Processamento paralelo com SMP: SMP é uma sigla que designa computadores com mais de um processador com as mesmas características, daí o termo *Symetric Multi Processor*. Os processadores compartilham o mesmo barramento e memória. Veja na Figura 2 (a) uma placa mãe modelo ASUS-CUV4X-DLS com dois slots para processadores PIII de 1000MHz. Esta placa mãe equipa os computadores do "cluster do LMPT" ilustrado na Figura 2 (b).

Requisitos: Os programas devem ser desenvolvidos com o uso de múltiplas threads ou múltiplos processos.

Vantagens: Relativamente fácil de programar.

Desvantagens: Requer máquinas com dois ou mais processadores (são máquinas caras).

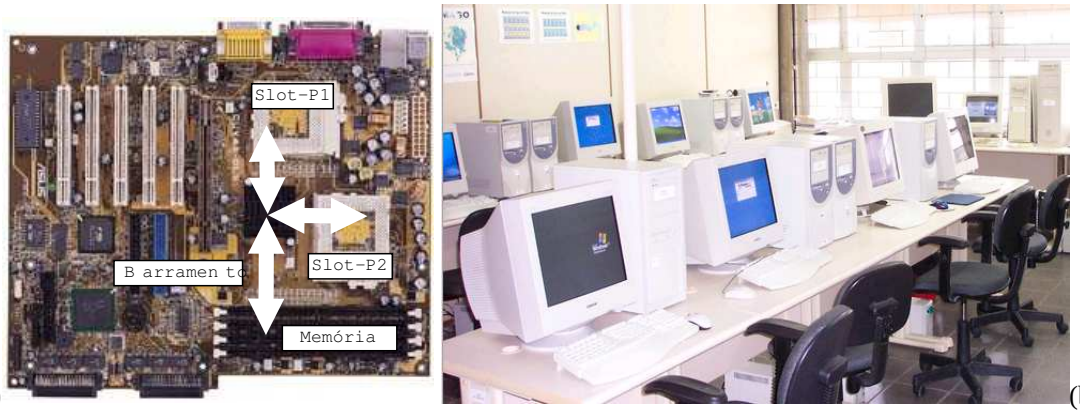
Processamento paralelo distribuído em um cluster com OpenMosix: O OpenMosix é um adendo ao kernel do GNU/Linux que adiciona ao mesmo capacidades de computação com cluster. Isto possibilita que as estações do cluster, baseadas em X86/Pentium/AMD, trabalhem de forma cooperativa, como sendo um único sistema. A migração dos processos entre as várias máquinas do cluster é automática, o que permite que programas antigos funcionem num ambiente de cluster com pouquíssimas alterações, Manika (1999). Entre as características do OpenMosix pode-se destacar o balanceamento dinâmico e inteligente de carga, uso com cluster heterogêneo, transparência, escabilidade, descentralização e autonomia dos nós, migração preemptiva dos processos com uso de algoritmos probabilísticos, comunicação entre núcleos eficiente, controle descentralizado. Observe que mesmo que o programa não seja desenvolvido utilizando múltiplos processos, o OpenMosix distribui os processos de uma máquina sobrecarregada para um cliente com pouca atividade. Outra característica do OpenMosix é que o gerenciamento da carga de cada máquina do cluster pode ser controlada automaticamente ou pelo usuário com programas utilitários. Veja ilustração do OpenMosixView na Figura 3. Veja uma apresentação do OpenMosix em (`http://www.ppgia.pucpr.br/~almendes/DIPC_MOSIX/ppframe.htm`).

Requisitos: Requer a recompilação do kernel com a inclusão do OpenMosix ou instalação de kernel em pacote (como os pacotes rpm do GNU/Linux/RedHat). O *The OpenMosix HOWTO*, disponível em `http://howto.ipng.be/openMosix-HOWTO/`, descreve em detalhes a instalação e uso do OpenMosix.

Vantagens: O trabalho de programação é reduzido, exigindo apenas a implementação dos mecanismos de troca de dados entre os diferentes processos. Otimização do uso das máquinas do cluster com a migração automática dos processos.

Desvantagens: Exige a recompilação do kernel.

[3]O OpenMosix é um adendo ao kernel do GNU/Linux que adiciona ao mesmo capacidades de computação com cluster.



(a) (b)

Figura 2: Em (a) placa mãe ASUS-CUV4X-DLS com slot para 2 processadores. Em (b) exemplo de um "*cluster de workstation*" (LMPT-Laboratório de Meios Porosos e Propriedades Termofísicas).

4. Bibliotecas para desenvolvimento de processamento paralelo[4]

Os programas podem ser desenvolvidos utilizando-se processos (4.1), threads (4.2), ou sistemas de troca de mensagens como PVM (4.3) e MPI (4.4).

4.1 Processos: De um modo geral, os computadores com sistemas operacionais multi-tarefa disponibilizam um conjunto de funções para divisão e compartilhamento do(s) processador(es) e da memória. Estes sistemas costumam disponibilizar chamadas ao kernel que possibilitam a criação de múltiplos processos. Se a máquina tem mais de um processador, o sistema operacional distribui os processos pelos processadores. No GNU/Linux e nas variantes do Unix, um processo pode ser clonado com a função fork. A comunicação entre os processos é feita de forma simplificada com o uso de pipes.

Requisitos: Requer o aprendizado do uso das instruções fork (para clonar processos) e pipe (para comunicação entre os processos). O que é um processo (sua estrutura, características e estados), como implementar e sincronizar os processos, como fica a alocação e o compartilhamento de recursos e dados. Veja detalhes em Hughes e Hughes (1997).

Vantagens: Pode ser utilizado com o OpenMosix, não sendo necessário acrescentar mecanismos de distribuição dos processos.

Desvantagens: Não permite o compartilhamento de memória.

4.2 Threads: Threads são múltiplos caminhos de execução que rodam concorrentemente na memória compartilhada e que compartilham os mesmos recursos e sinais do processo pai. Uma thread é um processo simplificado, mais leve ou "light", custa pouco para o sistema operacional, sendo fácil de criar, manter e gerenciar. Segundo Wilkinson e Allen (1999), o processamento paralelo com uso de threads é de mais alto nível que os mecanismos de troca de mensagens. Também tem um tempo de latência menor. O padrão internacional é o POSIX 1003.1C, o mesmo é compatível com outros sistemas operacionais como o Windows e o MAC_OS X.

Veja uma introdução sobre threads em (`http://centaurus.cs.umass.edu/~wagner/threads_html/tutorial.html`), um conjunto de links em (`http://pauillac.inria.fr/~xleroy/linuxthreads/`), descrições adicionais nas referências (Hughes e Hughes (1997); Wall (2001); Butenhof (1987); Maeney (1999)). Uma biblioteca orientada a objeto para o desenvolvimento de threads é disponível em (`http://www.gnu.org/directory/GNU/commoncpp.html`).

Requisitos: Requer o conhecimento da biblioteca de programação com threads (conhecida como *PThreads*). Conceitos adicionais como estrutura, características, estados e prioridades de uma thread. Como implementar a cooperação e sincronização das threads com uso de mutexes, semáforos e variáveis condicionais.

Vantagens: Em poucas palavras é o pacote definitivo para o desenvolvimento de programação em larga escala no GNU/Linux, Dietz (1998). É o mecanismo mais simples para implementação de processamento paralelo.

Desvantagens: Não pode ser utilizado com OpenMosix.

4.3 PVM (Parallel Virtual Machine); É a biblioteca mais utilizada para processamento distribuído, sendo o padrão de fato da indústria de software. O PVM se baseia em duas primitivas básicas, envie mensagem e receba mensagem. O usuário deve configurar as máquinas para que sejam o mais idênticas possível, facilitando a manutenção e estabelecendo uma relação de confiança entre elas. O usuário roda o gerenciador do PVM, adiciona máquinas ao cluster e depois simplesmente executa o programa feito usando as bibliotecas do PVM, Manika (1999). Veja mais detalhes em (Dietz (1998); Radajewski e Eadline (1998); Geist *et al.*(1994)).

[4]Existem várias outras opções de gerenciamento e controle da memória e dos processos (como o System V Shared Memory). De uma maneira geral, quanto maior a eficiência desejada, maior a complexidade dos modelos a serem utilizados. Você vai ter de aprender conceitos como atomicidade, volatilidade, travamento de memória, gerenciamento de cache.

Requisitos: Para o desenvolvimento dos programas é necessário conhecer a biblioteca PVM. É um sistema explicito, ou seja, cabe ao programador dividir as tarefas através da troca de mensagens.

Vantagens: Possibilita o uso do processamento distribuído e é o mais utilizado. Alguns programas de engenharia e matemática geram código automaticamente para o PVM.

Desvantagens: Não é mais o padrão. O desenvolvimento dos programas fica bem mais complicado quando comparado com threads.

Razões para usar PVM: PVM é mais antigo, usa o conceito de máquina virtual e tem maior interoperabilidade (o que possibilita a execução de uma mesma simulação em máquinas diferentes - exemplo SUN e DIGITAL. Veja o site `http://www.hpcvl.org/faqs/pvm/pvmGE.html#answer_2`.

4.4 MPI (Message Passing Interface): O MPI tem opções mais avançadas (que o PVM), como envio de mensagens broadcast (para todas as máquinas do cluster) e multicast (para um grupo específico de máquinas), assim como um melhor controle sobre o tratamento que cada mensagem terá ao ser recebida por outro ponto do cluster. É um método que inclui conceitos novos como rank (cada processo tem uma identificação única, crescente), group (conjunto ordenado de processos) e communicator (uma coleção de grupos), que permitem um gerenciamento mais complexo (e inteligente) do uso de cada máquina do cluster. A configuração do MPI depende da implementação utilizada e algumas delas chegam a instalar front-ends para compiladores em C e Fortran, mas a forma geral de uso é semelhante. Veja mais detalhes em (Dietz (1998); Wilkinson e Allen (1999); Pacheco (1996)).

Requisitos: Requer o conhecimento de um sistema bastante complexo de troca de mensagens, o MPI.

Vantagens: É o novo padrão para processamento distribuído, embora ainda seja menos utilizado que o PVM.

Desvantagens: Na prática significa aprender uma nova linguagem de programação. É um padrão da indústria com várias implementações individuais. É complicado.

Razões para usar MPI: MPI tem maior performance, maior controle dos recursos e é mais fácil de trabalhar quando comparado com PVM. Se o programa vai ser executado em uma mesma arquitetura (que suporte LAM) dê preferência ao MPI. Veja outras razões no site `http://www.lam-mpi.org/mpi/mpi_top10.php`.

5. Comparação dos tempos de processamento

A Tabela 1 apresenta uma comparação do tempo de processamento na determinação do valor de pi utilizando um algoritmo em C++ com multi-threadings e com multi-processing.

Os computadores utilizados para os testes tem placa mãe ASUS-CUV4X-DLS com 2 processadores PIII de 1000MHz, 2GB de memória ram, placa de rede Intel EtherExpress Pro 100b de 100Mbs, e estão conectados a um switch 3com SuperStack III de 100Mbs usando cabos de par trançado. O sistema operacional utilizado é GNU/Linux com OpenMosix e kernel 2.4.19. O compilador é o gcc 2.96. O cluster foi montado com 3 computadores (6 processadores).

Com 1 thread o tempo de processamento ficou em 1m4.850s, o uso de 2 threads é vantajoso (0m32.443s). Como esperado, o uso de 4 threads implica em um tempo de processamento maior (0m32.602s), pois todas as threads são executadas no mesmo computador (o OpenMosix não distribui processos que executem threads).

O uso de múltiplos processos com o OpenMosix tem o comportamento esperado, o que pode ser verificado através do programa de monitoramento que acompanha o OpenMosix (Figura 3). A simulação com 2 processos foi toda executada no mesmo computador, não houve redistribuição dos processos, e o tempo de processamento ficou em 0m22.963s. Quando se usa 4 processos, o OpenMosix distribui os processos pelas máquinas do cluster, e o tempo de processamento ficou em 0m12.592s. Com seis processos o tempo de processamento ficou em 0m10.090s. De um modo geral, o tempo de processamento pode ser estimado pela relação tempoNormal/númeroProcessos com perdas em função da redistribuição dos processos.

Tabela 1: Comparação dos tempos de processamento utilizando-se múltiplas-threads e múltiplos-processos em um cluster de 3 nós (6 processadores).

número threads	tempo	número processos	tempo
1	1m4.850s	2	0m22.963s
2	0m32.443s	4	0m12.592s
4	0m32.602s	6	0m10.090s

Como se aprofundar (leituras aconselhadas) Iniciar com os HOWTOS: i) *Beowulf Howto*, um como fazer que descreve uma série de conceitos e técnicas para implementação de um cluster (`http://www.sci.usq.edu.au/staff/jacek/beowulf/BDP`). ii) *Linux Parallel Processing HOWTO*, descreve os diferentes tipos de processamento paralelo e conceitos básicos de cluster (`http://yara.ecn.purdue.edu/~pplinux/PPHOWTO/pphowto.html`). iii) *The OpenMosix HOWTO*, descreve em detalhes a instalação e uso do OpenMosix. A home page do projeto BEOWULF, em (`http://www.beowulf.org`) contém diversas informações adicionais. As principais bibliotecas são descritas em *PVM - Parallel Virtual Machine* (`http://www.epm.ornl.gov/pvm/pvm_home.html`), *LAM/MPI (Local Area Multicomputer / Message Passing Interface* (`http://www.mpi.nd.edu/lam`), threads (`http://www.humanfactor.com/pthreads/`).

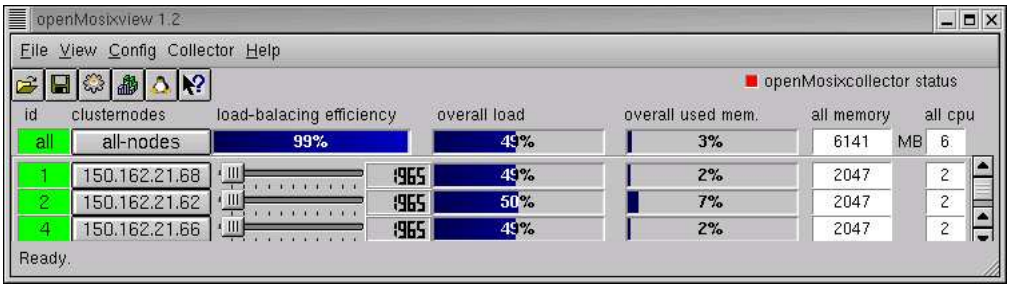


Figura 3: O openMosixView, programa de monitoramento dos diversos nós do cluster (carga, memória, processadores).

6. Conclusões

O uso do processamento paralelo em um cluster de computadores é uma ferramenta extremamente poderosa, possibilitando o desenvolvimento de simulações avançadas em engenharia de rochas reservatório em sistemas de baixo custo. Como visto, os computadores podem ser utilizados para processamento comum de dia e para processamento pesado a noite e nos finais de semana, aproveitando melhor o parque de máquinas instaladas.

Os pesquisadores precisam aprender os conceitos básicos de processamento paralelo e as diferentes formas de distribuição do processamento (processos, threads, PVM, MPI). Os conceitos básicos destes sistemas e referências externas foram apresentados.

O mecanismo mais fácil de desenvolver processamento paralelo envolve a utilização de múltiplas-threads e a seguir múltiplos-processos, sendo aconselhável iniciar com estes mecanismos. Posteriormente, pode-se trabalhar com bibliotecas especializadas como PVM e MPI. Em ambos os casos procure utilizar uma biblioteca padrão e multi-plataforma.

7. Referências

ADLER, P. M., JACQUIN, C. G., QUIBLIER, J. A. Flow in simulated porous media. *Int. J. Multiphase Flow*, v16, p.691-712. 1990.

BUENO, A. D., MAGNANI, F. S., PHILIPPI, P. C. Método para determinação da permeabilidade relativa de rochas reservatório de petróleo através da análise de imagens reconstruídas. page 12, *IX Congresso Brasileiro de Engenharia e Ciências Térmicas - ENCIT 2002*. Caxambú - MG - Brasil. 2002a.

BUENO, A. D., PHILIPPI, P. C. Modelo do grafo de conexão serial para determinação da permeabilidade de rochas reservatório de petróleo. *IX Congresso Brasileiro de Engenharia e Ciências Térmicas - ENCIT 2002*, Caxambú - MG - Brasil. p.1-12. 2002.

BUENO, A. D., SANTOS, L. O. D., FERNANDES, C. P., PHILIPPI, P. C. Reconstrução tridimensional da micro-estrutura de rochas reservatório a partir de lâminas finas. Caxambú - MG - Brasil. *IX Congresso Brasileiro de Engenharia e Ciências Térmicas - ENCIT 2002*. p.1-12. 2002.

BUTENHOF, D. R. Programming with POSIX(R) Threads. *Addison-Wesley*. 1987.

COSTER, M. CHERMANT, J. L. Precis D Analyse D Images, volume 1. *PRESSES DU CNRS*, Paris. 1989.

DANIEL RIDGE, DONALD BECKER, P. M. T. S. B. MERKEY, P. Harnessing the power of parallelism in a pile-of-pcs. *IEEE Aerospace*. 1997.

DIETZ, H. Linux Parallel Processing HOWTO. `http://yara.ecn.purdue.edu/pplinux/PPHOWTO/pphowto.html`. 1998.

DONALD J. BECKER, THOMAS STERLING, D. S. B. F. K. O. Communication overhead for space science applications on the beowulf parallel workstation. *High Performance Distributed Computing*. 1995.

DONALD J. BECKER, THOMAS STERLING, D. S. J. E. D. U. A. R. C. V. P. . Beowulf: A parallel workstation for scientific computation. *International Conference on Parallel Processing.* 1995b.

GEIST, A., BEGUELIN, A., DONGARRA, J. PVM: Parallel Virtual Machine. MIT Press. 1994.

GOMES, J. VELHO, L. Computação Gráfica :Imagem. *INPA-SBM*, Rio de Janeiro, 1rd edition. ISBN. 1994.

GONZALES, R. WOODS, R. Digital Image Processing. Addison-Wesley, 1rd edition. ISBN. 1993.

HUGHS, C. HUGHES, T. Object Oriented Multithreading using C++: architectures components, v1. *John Wiley Sons*, 2 ed. 1997.

MANIKA, G. W. Super-Computador a Preço de Banana, volume 2. Revista do Linux. 1999.

MASNEY, B. Introdutcion to multi-thread programming. *Linux Journal*. v61, maio, 1999.

PACHECO, P. Parallel Programming With MPI. *Morgan Kaufmann Publishers*. 1996.

RADAJEWSKI, J. EADLINE, D. Beowulf HOWTO. http://www.sci.usq.edu.au/staff/jacek/beowulf/BDP. 1998. SANTOS, L. O. E., PHILIPPI, P. C., DAMIANI, M. C., FERNANDES, C. P. Using three-dimensional reconstructed microstructures for predicting intrinsic permeability of reservoir- rocks based on a boolean lattice gas method. *Journal of Petroleum Science and Engineering-JPSE*. v35, p.109-124.2002.

WAGNER, T. TOWSLEY, D. Getting started with posix threads. *University of Massachusetts at Amherst*. 1995.

WALL, K. Linux Programming Unleashed, volume 1. *SAMS*, 2 edition. 2001.

WILKINSON, B. ALLEN, C. M. Parallel Programming: Techniques and Applications Using Workstation and Parallel Computers. *Prentice Hall*. 1999.